IN THE CLAIMS

1. [currently amended] A process comprising:

validating a formal language specification written in a formal language which has predetermined rules of syntax and semantics, said formal language specification defining a computer program to be automatically written;

~~for~~ automatically translating each element of a ~~a~~ formal language specification defining an object model, a functional model, a dynamic model and a presentation model, which taken together define the requirements of the program to be automatically written, ~~written in a formal language defining a full and complete computer program to be automatically written by a computer~~ into a full and complete ~~source code~~ computer program which ~~that can be compiled into a complete, executable program which can execute by itself on a computer and~~ needs no additional third party source code or source code from existing components or code libraries to be compiled with it to make said computer program complete ~~executable program~~ and which implements the requirements of said formal language specification, said formal language specification defining at least an identification function for every class, and at least a valuation for every variable attribute ~~classes of objects having attributes, services and relationships with other classes,~~ said translating step comprising the following steps:

using a computer, automatically write computer code that will request user name and password, receive any responses and authenticate the user;

using a computer, automatically write computer code that will determine this user's privilege level and query said formal language specification and determine all object attributes said ~~this~~ user has privilege to see and ~~query and~~ all services said ~~this~~ user can invoke;

using a computer, automatically write computer code which queries said

formal language specification for all services of all classes that any authorized user may invoke and identifies an object server which will implement each said service;

using a computer, automatically write code that will retrieve service arguments for all services ~~from one or more of a user, an object server, and another process, as appropriate~~;

using a computer, automatically write code that controls a computer to display means by which and entity can invoke a service, and which receives input to invoke a particular service and responds by sending a message to the appropriate object server to invoke the service, said message including the necessary arguments for the service to execute;

using a computer, automatically write code that implements an object server for every service, each of which first checks to verify that state transitions are valid and make sense for the current state of objects of which the object server ~~service~~ will be altering the state;

using a computer, automatically write code for every object server that verifies preconditions are satisfied before making state transitions of any objects the states of which are acted upon by the object server;

using a computer, automatically write code to make all valuation calculations required by said formal language specification of each object server;

using a computer, automatically write code to verify that integrity constraints specified in said formal language specification on the values of attributes of objects have been satisfied after execution of a service and take action if said integrity constraints are not satisfied; and

using a computer, automatically write code for every object server to test

trigger relationships specified in said formal language specification after execution of a service and carry out appropriate action if a trigger event has occurred.

2. [currently amended] An apparatus for automatically translating a formal language specification defining an object model, a functional model, a dynamic model and a presentation model, which taken together define the requirements of a computer program to be automatically written, said formal language specification being written in a formal language which has predefined rules of grammar, ~~said formal specification defining a computer program to be automatically written by a computer~~, said translating acting to convert said formal language specification into a computer program that implements the requirements of said formal language specification, said formal language specification defining at least an identification function for every class and at least a valuation for every variable attribute, ~~classes of objects having attributes, services and relationships with other classes,~~ comprising:

a computer programmed with an operating system and one or more other programs to cooperate with said operating system to control said computer to perform the following functions:

A) using said predetermined rules of grammar to validate said formal language specification to ensure that said formal language specification is complete and correct;

B) automatically write computer code that will request user name and password, receive any responses and authenticate the user;

C) automatically write computer code that will determine a user's privilege level for a user identified by user name and password entered in response to the

query caused by the code written in step B, and query said formal specification and determine all object attributes said user has privilege to see and query and all services said user can invoke;

D) automatically write computer code which queries said formal specification for all services of all classes that any authorized user may invoke and identifies an object server which will implement said service;

E) automatically write code that will retrieve service arguments for all services ~~from one or more of a user, another object server, and another process, as appropriate~~;

F) automatically write code that displays one or more user interface tools which can be used to invoke a service, and which receives input to invoke a particular service and which responds by sending a message to the appropriate object server to invoke said service, said message including the necessary arguments for said service to execute;

G) automatically write code that implements an object server for every service, each of which first checks to verify that state transitions are valid and make sense for the current state of objects the object service will be altering the state of;

H) automatically write code for every object server that verifies preconditions are satisfied before making state transitions of any objects the states of which are acted upon by said object server;

I) automatically write code to make all valuation calculations required by said specification of each object server;

J) automatically write code to verify that integrity constraints specified in said formal specification on the values of attributes of objects have been satisfied

after execution of a service and take action if said integrity constraints are not satisfied; and

K) automatically write code for every object server to test trigger relationships specified in said formal specification after execution of a service and carry out appropriate action if a trigger event has occurred.

3. [currently amended] A computer-readable medium containing instructions for controlling a computer to automatically translate a formal language specification defining an object model, a functional model, a dynamic model and a presentation model, which taken together define the requirements of a computer program to be automatically written, ~~defining a computer program to be automatically written by a computer into a computer program that implements the requirements of said formal specification,~~ said formal language specification defining at least an identification function for every class and at least a valuation for every variable attribute ~~classes of objects having attributes, services and relationships with other classes,~~ said formal language specification written in a formal language having predefined rules of grammar, by:

validating a formal language specification written in a formal language which has predetermined rules of syntax and semantics, said validating accomplished using said predetermined rules of syntax and semantics to ensure said formal language specification is complete and correct;

automatically writing computer code that will request user name and password, receive any responses and authenticate the user;

automatically writing computer code that will determine a user's privilege level and query said formal specification and determine all object attributes said user has privilege to see and all services said user can invoke;

automatically writing computer code which queries said formal specification for all services of all classes that any authorized user may invoke and identifies an object server which will implement said service;

automatically writing computer code that will retrieve service arguments for all services ~~from one or more of a user, an object server, and another process, as appropriate~~;

automatically write code that displays menus options, icons or creates any other means by which a user or another process can invoke a service, and which receives input to invoke a particular service and responds by sending a message to the appropriate object server to invoke the service, said message including the necessary arguments for the service to execute;

automatically writing code that implements an object server for every service, each of which first checks to verify that state transitions are valid and make sense for the current state of objects the object service will be altering the state of;

automatically write code for every object server that verifies preconditions are satisfied before making state transitions of any objects the states of which are acted upon by the object server;

automatically write code to make all valuation calculations required by said formal specification of each object server;

automatically write code to verify that integrity constraints specified in said formal specification on the values of attributes of objects have been satisfied after execution of a service and take action if said integrity constraints are not satisfied; and

automatically write code for every object server to test trigger relationships specified in said formal specification after execution of a service and carry out appropriate action if a trigger event has occurred.

Please add the following new claims

4. [new] An apparatus for automatically translating a Formal Language Specification written in any formal language defining a full and complete Conceptual Model of a desired computer program to be automatically generated into a full and complete source code which implements said desired computer program, comprising:

a computer programmed with an operating system and one or more other programs to cooperate with said operating system to control said computer to perform the following functions:

reading all said primitives in said Formal Language Specification in any order;

in any order, generating one or more methods comprised of computer code which can control a computer to perform the following functions in an order determined by an execution model:

determining if said Formal Language Specification requires user authentication, and, if so, automatically writing computer code that will request user name and password, receive any responses and authenticate the user;

determining if said Formal Language Specification requires determining a user privilege level, and, if so, automatically writing computer code that will determine a user's privilege level and query said Formal Language Specification and determine all object attributes said user has privilege to see and determine all services this user can invoke;

determining if said Formal Language Specification defines

services, and, if so, automatically writing computer code which queries said Formal Language Specification to determine all services that the authenticated user may invoke and which are defined in said Formal Language Specification for all classes of objects said authenticated user will be able to view and automatically writing an object server for each said service which will implement said service upon receipt of a service invocation message, each of said object server containing code which will perform the following functions in the following order upon receipt of a service invocation message:

verify that one or more state transitions can be validly made for the current state of any object(s) of which said object server will be altering the state before actually altering the state of said object(s);

verify that any preconditions of a service implemented by an object server are satisfied before said object server will act upon said one or more objects to make state transitions thereof in carrying out said service,

ignoring said service invocation message if either said one or more state transitions cannot be validly made for the current state of any objects upon which said object server will be acting or any said precondition is not satisfied;

if all said transitions can be validly made on said one or more objects upon which said object server will act and if all said preconditions are satisfied, make all valuation calculations of said object server required by said Formal Language Specification;

verify that service execution by said object server did not result in violation of one or more integrity constraints specified in said Formal Language Specification on the values of attributes of objects affected by execution of said service implemented by said object server, and take corrective action if one or more of said integrity constraints are not satisfied; and

after a valid change of state of an object acted upon by said object server occurs, test trigger relationships or condition-action rules specified in said Formal Language Specification and, if any trigger event is satisfied, triggering a service specified in said condition-action rule or trigger relationship.

5. [New] The apparatus of claim 4 wherein said computer is further programmed to perform the following process:

presenting graphical user interface tools which can be invoked by a user to define primitives of said Formal Language Specification which detail an object model, a dynamic model including a state transition diagram for every class and an object interaction diagram for every trigger and for every global transaction, a functional model defining the dynamic formulas of said evaluations which define the effect of events on variable of attributes defined for objects of said object model, and a presentation model which defines the desired user interface of said desired computer program.

6. [New] The apparatus of claim 4 wherein said computer is further programmed to perform the following process:

displaying a CASE tool with graphical user interface tools which can be invoked by a user to define primitives of said Formal Language Specification which detail an object model, a dynamic model including a state transition diagram for every class and an object interaction diagram for every trigger and for every global transaction, a functional model defining the dynamic formulas of said evaluations which define the effect of events on variable attributes of attributes defined for objects of said object model, and a presentation model which defines the desired user interface of said desired computer program, said CASE tool automatically converting said primitives to a formal language specification written in a formal language with predefined rules of syntax and semantics; and

validating said formal language specification to make sure it is complete in that there is no missing information and all required properties of said Conceptual Model are defined and have a value, and correct in that information introduced in said Conceptual Model is syntactically and semantically consistent and not ambiguous.

7. [New] A computer-readable medium containing instructions for controlling a computer to automatically translate a formal language specification defining a desired computer program into a full and complete computer program that implements the requirements of said formal language specification, said instructions defining one or more methods which when executed by a computer cause said computer to perform the following functions:

reading all said primitives in said Formal Language Specification in any order;

in any order, generating one or more methods comprised of computer

code which can control a computer to perform the following functions in an order determined by an execution model:

determining if said Formal Language Specification requires user authentication, and, if so, automatically writing computer code that will request user name and password, receive any responses and authenticate the user;

determining if said Formal Language Specification requires determining a user privilege level, and, if so, automatically writing computer code that will determine a user's privilege level and query said Formal Language Specification and determine all object attributes said user has privilege to see and determine all services this user can invoke;

determining if said Formal Language Specification defines services, and, if so, automatically writing computer code which queries said Formal Language Specification to determine all services that the authenticated user may invoke and which are defined in said Formal Language Specification for all classes of objects said authenticated user will be able to view and automatically writing an object server for each said service which will implement said service upon receipt of a service invocation message, each of said object server containing code which will perform the following functions in the following order upon receipt of a service invocation message:

verify that one or more state transitions can be validly made for the current state of any object(s) of which said object server will be altering the state before actually altering the state of said object(s);

verify that any preconditions of a service implemented by an object server are satisfied before said object server will act upon said one or more objects to make state transitions thereof in carrying out said service,

ignoring said service invocation message if either said one or more state transitions cannot be validly made for the current state of any objects upon which said object server will be acting or any said precondition is not satisfied;

if all said transitions can be validly made on said one or more objects upon which said object server will act and if all said preconditions are satisfied, make all valuation calculations of said object server required by said Formal Language Specification;

verify that service execution by said object server did not result in violation of one or more integrity constraints specified in said Formal Language Specification on the values of attributes of objects affected by execution of said service implemented by said object server, and take corrective action if one or more of said integrity constraints are not satisfied; and

after a valid change of state of an object acted upon by said object server occurs, test trigger relationships or condition-action rules specified in said Formal Language Specification and, if any trigger event is satisfied, triggering a service specified in said condition-action rule or trigger relationship.

8. [New] The computer-readable medium of claim 7 further storing computer

instructions which, when executed, control a computer to carry out the following process:

presenting graphical user interface tools which can be invoked by a user to define primitives of said Formal Language Specification which detail an object model, a dynamic model including a state transition diagram for every class and an object interaction diagram for every trigger and for every global transaction, a functional model defining the dynamic formulas of said evaluations which define the effect of events on variable attributes of attributes defined for objects of said object model, and a presentation model which defines the desired user interface of said desired computer program.

9. [New] The computer-readable medium of claim 7 further storing computer instructions which, when executed, control a computer to carry out the following process:

displaying a CASE tool with graphical user interface tools which can be invoked by a user to define primitives of said Formal Language Specification which detail an object model, a dynamic model including a state transition diagram for every class and an object interaction diagram for every trigger and for every global transaction, a functional model defining the dynamic formulas of said evaluations which define the effect of events on variable attributes defined for objects of said object model, and a presentation model which defines the desired user interface of said desired computer program, said CASE tool automatically converting said primitives to a formal language specification written in a formal language with predefined rules of syntax and semantics; and

validating said formal language specification to make sure it is complete in

that there is no missing information and all required properties of said Conceptual Model are defined and have a value, and correct in that information introduced in said Conceptual Model is syntactically and semantically consistent and not ambiguous.

10. [new] A process for automatically translating a Formal Language Specification written in any formal language defining a full and complete Conceptual Model of a desired computer program to be automatically generated into a full and complete source code which implements said desired computer program, comprising:

reading all said primitives in said Formal Language Specification in any order;

in any order, generating one or more methods which perform the following functions in an order determined by an execution model:

determining if said Formal Language Specification requires user authentication, and, if so, automatically writing computer code that will request user name and password, receive any responses and authenticate the user;

determining if said Formal Language Specification requires determining a user privilege level, and, if so, automatically writing computer code that will determine a user's privilege level and query said Formal Language Specification and determine all object attributes said user has privilege to see and determine all services this user can invoke;

determining if said Formal Language Specification defines services, and, if so, automatically writing computer code which queries said Formal Language Specification to determine all services that the

authenticated user may invoke and which are defined in said Formal Language Specification for all classes of objects said authenticated user will be able to view and automatically writing an object server for each said service which will implement said service upon receipt of a service invocation message, each of said object server containing code which will perform the following functions in the following order upon receipt of a service invocation message:

verify that one or more state transitions can be validly made for the current state of any object(s) of which said object server will be altering the state before actually altering the state of said object(s);

verify that any preconditions of a service implemented by an object server are satisfied before said object server will act upon said one or more objects to make state transitions thereof in carrying out said service,

ignoring said service invocation message if either said one or more state transitions cannot be validly made for the current state of any objects upon which said object server will be acting or any said precondition is not satisfied;

if all said transitions can be validly made on said one or more objects upon which said object server will act and if all said preconditions are satisfied, make all valuation calculations of said object server required by said Formal Language Specification;

verify that service execution by said object server did not result in violation of one or more integrity constraints specified in

said Formal Language Specification on the values of attributes of objects affected by execution of said service implemented by said object server, and take corrective action if one or more of said integrity constraints are not satisfied; and

after a valid change of state of an object acted upon by said object server occurs, test trigger relationships or condition-action rules specified in said Formal Language Specification and, if any trigger event is satisfied, triggering a service specified in said condition-action rule or trigger relationship.

11. [New] The process of claim 10 further comprising the steps of:

presenting graphical user interface tools which can be invoked by a user to define primitives of said Formal Language Specification which detail an object model, a dynamic model including a state transition diagram for every class and an object interaction diagram for every trigger and for every global transaction, a functional model defining the dynamic formulas of said evaluations which define the effect of events on variable attributes of attributes defined for objects of said object model, and a presentation model which defines the desired user interface of said desired computer program.

12. [New] The process of claim 10 further comprising the steps of:

displaying a CASE tool with graphical user interface tools which can be invoked by a user to define primitives of said Formal Language Specification which detail an object model, a dynamic model including a state transition diagram for every class and an object interaction diagram for every trigger and for every

global transaction, a functional model defining the dynamic formulas of said evaluations which define the effect of events on variable attributes of attributes defined for objects of said object model, and a presentation model which defines the desired user interface of said desired computer program, said CASE tool automatically converting said primitives to a formal language specification written in a formal language with predefined rules of syntax and semantics; and

validating said formal language specification to make sure it is complete in that there is no missing information and all required properties of said Conceptual Model are defined and have a value, and correct in that information introduced in said Conceptual Model is syntactically and semantically consistent and not ambiguous.

13. [New-page 72 of cip 3] A process for converting a Formal Language Specification encoding a Conceptual Model which defines desired system logic and a desired user interface of a desired computer program into source code which encodes said desired computer program, comprising:

validating said Formal Language Specification to ensure it is complete and correct;

retrieving predetermined information from said Formal Language Specification encoding a Conceptual Model which defines one or more classes of objects in an object model, a dynamic model which specifies the behavior of each object in response to services, triggers and global transactions as represented by a state transition diagram for every class and an object interaction diagram for every trigger and for every global transaction, and a functional model which defines the semantics of any change of each object's state as a consequence of an event occurrence by specifying for each class one or more mathematical or logical formulas which define how one or more variable

attributes of said class will have their values changed when one or more specified events of said class occurs meaning one or more services of said class is executed;

using predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a presentation tier of said desired computer program;

using predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a persistence tier, database of data structure of said desired computer program;

using said predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a middle tier of said desired program which communicates with said presentation tier in the manner defined below, said middle tier written by automatically generating at least the following component instances for each class defined in said Formal Language Specification:

a server component instance including a method to implement each service present in a signature of said class and one or more methods to receive requests from said presentation tier that relate to execution of services of said class;

a query component instance including a method for implementing queries to extract information from said persistence tier relating to objects within said class and a method to receive and process requests from said presentation tier to query said persistence tier;

an executive component instance including one or more methods to receive and process a request from said server component or another executive component to execute a service in said class and carry out the following functions:

verify the existence and validity for a requested server component;

create a copy of a requested server component instance in memory and access said persistence tier using said query component to retrieve values of constant and variable attributes of said server component;

validate state transitions for said requested service and a present state for a server component instance;

verify the satisfaction of preconditions specified in said Formal Language Specification of said requested service;

changing the state of said server component instance to a new state by modifying a value of a variable attribute of said server component instance by performing all valuations specified in said functional model affected by said requested service;

validating said new state by verifying said new state does not violate static or dynamic restrictions specified in said Formal Language Specification;

check trigger conditions established in said Formal Language Specification to determine if said new state causes any trigger to occur, and, if so, which actions should be carried out;

communicate with said persistence tier to access or store all attributes of said server component instance.

14. [New - p. 40 of clean copy of substitute specification enclosed herewith] A process for validating a formal language specification, comprising the steps:

A) checking said formal language specification to ensure that it is complete in that all required properties of a Conceptual Model embodied in said formal language specification are defined and have a valid value;

B) using rules of grammar of whatever formal language said formal language specification is written in, checking said formal language specification to ensure it is correct in that it is syntactically and semantically correct and not ambiguous.

15. [new] The process of claim 14 further comprising the step of presenting a request for said missing information via a mechanism of a user interface if any information is missing or presenting a request via a user interface mechanism to correct any syntactic or semantic error or clarify any ambiguity discovered during said validation process.

16. [new] The process of claim 14 further comprising the step of doing a partial validation each time an element is added to said formal language specification to check for completeness and correctness and mark the portion of said formal language specification just added as invalid if an error is found so that a request to correct said error can be presented later when a full validation of said formal language specification is requested.

17. [new] The process of claim 14 wherein step A comprises checking to ensure that all elements in said Conceptual Model have a set of properties that exist and have a valid value and wherein step B comprises using a predetermined process and grammar for every type of formula in said Conceptual Model to ensure each is syntactically and

semantically correct.

18. [new] The process of claim 14 wherein step A comprises performing strict validation on some element properties and flexible validation of other element properties of said Conceptual Model, said strict validation of a property defined as requiring a full definition and valid value for a property, and flexible validation defined as allowing a property to be incomplete or have an invalid value during a process of inputting elements which define said Conceptual Model, and wherein the following table defines which elements and properties have strict or flexible validations:

Element	Property	Subproperty	Validation Type
class			
	name		strict
	ID function		flexible
	attributes (at least one)		flexible
	services (at least a Create service)		flexible
	static and dynamic integrity constraints	their formulas	strict
attribute			
	name		strict

	type (constant, variable, derived)		strict
	data-type (real, integer, etc.)		strict
	default value		strict
	size		strict
	request in creation service		strict
	null value allowed		strict
	evaluations (variable attributes)		flexible
	derivation formula (derived attributes)		flexible
Evaluation			
	one variable attribute of a class		strict
	one service of the same class		strict
	condition		strict

	formula of evaluation		strict
derivation			
	formula		strict
	condition (optional)		strict
service			
	name		strict
	arguments		
		argument's name	strict
		data type	strict
		default value (optional)	strict
		null value	strict
		size (if proceeds)	strict
		formula of transaction	flexible
preconditions of an action			
	formula		strict

		agents affected by condition	strict
relationship: aggregation			
	related classes (component and composite)		strict
	relationship name		strict
	both directions: role names		strict
	cardinality		strict
	inclusive or referential		strict
	dynamic		strict
	clause "group by" (optional)		strict
	insertion and deletion events (if proceed)		strict
relationship: inheritance			

	related classes (parent & child)		strict
	temporal (versus permanent)		strict
	specialization condition or events		strict
relationship: agent			
	agent class and service allowed to activate		strict
state transition diagram			
	all states of classes (three at least)		flexible
state in state transition diagram			
	name		strict
transition in state transition diagram			
	estate of origin		strict

	estate of destination		strict
	service of class		strict
		control condition (optional)	strict
trigger			
	condition		strict
	class or instance of destination		strict
	target (self, object, class)		strict
	activated service		strict
	service arguments' initialization (optional)		
		argument values	strict
global interactions			
	name		strict
	formula		strict
user exit functions			

	name		strict
	return data-type		strict
	arguments, (optional)		
	argument's name		strict
	argument's data-type		strict.

19. [new] The process of claim 14 wherein step B comprises validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas, and wherein a predetermined process for validation and a set of rules of grammar exist for each type of formula and a validation process and a set of rules appropriate for the type of formula being validated is used for validation of each formula.

20. [new] The process of claim 14 wherein step B comprises validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments

inheritance: specialization conditions	
static and dynamic integrity constraints	
derivations and valuations	
	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

and further comprising the steps of:

presenting dialog boxes via a user interface by which a user may enter said formulas during a process of defining said Conceptual Model;

and wherein step B is performed by preventing a user from leaving a dialog box being used to define a formula until said formula being defined is syntactically and semantically correct.

21. [new] The process of claim 14 wherein steps A and B are performed by at least checking to ensure that every formula is syntactically correct, every class has an identification function and has a creation event and a destroy event, every triggering formula is semantically correct, every name of an aggregation is unique in the scheme of

said Conceptual Model, every derived attribute has at least a derivation formula, every service has an agent or server declared to execute it.

22. [new] The process of claim 14 further comprising performing steps A and B each time a user working on defining said Conceptual Model makes a change which may invalidate one or more formulas, but wherein predetermined formulas are allowed to be temporarily incorrect so that the user can review them at a later time when a full validation process is performed after the user is done defining the Conceptual Model.

23. [new] The process of claim 14 further comprising the step of using a computer to automatically translate said formal language specification into computer code after steps A and B and been completed and all formulas are syntactically and semantically complete and correct.

24. [new] The process of claim 14 further comprising the steps:

presenting user interface tools by which a user may define said Conceptual Model and make changes thereto;

checking all affected formulas each time a change is made to said Conceptual Model;

if the change affects a strictly validated property, then the change is rejected if the property is not given a valid value, otherwise the change is accepted;

if the change affects a property which is not strictly validated, then the user is informed should any error arise, but allowed to do the modification if he or she wishes;

if there are no affected formulas, modifying the Conceptual Model as specified by the user.

25. [new] The process of claim 14 wherein user interface defining portions of said formal language specification are validated by:

verifying that any patterns defined by said user are acceptable user interface patterns with no essential information missing;

attributes used in filters specified as part of said user interface are visible from a definition class;

attributes used in order criteria are visible from a definition class;

any formula in a filter is syntactically and semantically correct and uses only terms defined in said Conceptual Model;

any action selection pattern uses as final actions object defined in said Conceptual Model;

any set of dependency patterns are terminal and have confluence; and warnings are displayed to said user if any pattern is defined but not used or if an instance pattern is duplicated.

26. [new] An apparatus comprising a computer programmed with an operating system and a validation program that cooperates with said operating system to control said computer to perform the following functions of a validation process:

A) check a formal language specification for completeness by checking to ensure said formal language specification has no missing information which is needed to detail the requirements of a desired computer program modelled by said formal language specification;

b) cooperate with said operating system to ensure said formal language specification is correct by checking each statement in said formal language specification according to rules of grammar of whatever formal language in which said formal language specification is written to ensure that each statement is syntactically and semantically correct and not ambiguous so as to ensure that all properties of elements in a Conceptual Model encoded in said formal language specification have a value which is valid and to ensure that all formulas in said Conceptual Model have correct syntax and meaning.

27. [new] The apparatus of claim 26 wherein said validation program controls said computer to perform the following additional function:

presenting a request for said missing information via a mechanism of a user interface if any information is missing or presenting a request via a user interface mechanism to correct any syntactic or semantic error or clarify any ambiguity discovered during said validation process.

28. [new] The apparatus of claim 26 wherein said validation program controls said computer to perform the following additional function:

doing a partial validation each time an element is added to said formal language specification to check for completeness and correctness and mark the portion of said formal language specification just added as invalid if an error is found so that a request to correct said error can be presented later when a full validation of said formal language specification is requested.

29. [new] The apparatus of claim 26 wherein said validation program controls

said computer to perform the following additional function:

checking to ensure that all elements in said Conceptual Model have a set of properties that exist and have a valid value;

performing step B by using a predetermined process and grammar for every type of formula in said Conceptual Model to ensure each is syntactically and semantically correct.

30. [new] The apparatus of claim 26 wherein said validation program controls said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas, and wherein a predetermined process for validation and a set of rules of grammar exist for each type of formula and a validation process and a set of rules appropriate for the type of formula being validated is used for validation of each formula.

31. [new] The apparatus of claim 26 wherein said validation program controls said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	

static and dynamic integrity constraints	
derivations and valuations	
	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

and wherein said validation program is further structured to control said computer to perform the steps of:

presenting dialog boxes via a user interface by which a user may enter said formulas during a process of defining said Conceptual Model;

and wherein step B is performed by preventing a user from leaving a dialog box being used to define a formula until said formula being defined is syntactically and semantically correct.

32. [new] The apparatus of claim 26 wherein said validation program controls said computer to perform the following steps:

presenting user interface tools by which a user may define said Conceptual Model and make changes thereto;

checking all affected formulas each time a change is made to said Conceptual Model;

if the change affects a strictly validated property, then the change is rejected if the property is not given a valid value, otherwise the change is accepted;

if the change affects a property which is not strictly validated, then the user is informed should any error arise, but allowed to do the modification if he or she wishes;

if there are no affected formulas, modifying the Conceptual Model as specified by the user.

33. [new] An computer readable medium having stored thereon computer-readable instructions which when executed by a computer implement a validation program to control said computer to perform the following functions of a validation process:

A) check a formal language specification for completeness by checking to ensure said formal language specification has no missing information which is needed to detail the requirements of a desired computer program modelled by said formal language specification;

b) cooperate with said operating system to ensure said formal language specification is correct by checking each statement in said formal language specification according to rules of grammar of whatever formal language in which said formal language specification is written to ensure that each statement is syntactically and semantically correct and not ambiguous so as to ensure that all properties of elements in a Conceptual Model encoded in said formal language

specification have a value which is valid and to ensure that all formulas in said Conceptual Model have correct syntax and meaning.

34. [new] The computer-readable medium of claim 33 wherein said validation program is further structured to control said computer to perform the following additional function:

presenting a request for said missing information via a mechanism of a user interface if any information is missing or presenting a request via a user interface mechanism to correct any syntactic or semantic error or clarify any ambiguity discovered during said validation process.

35. [new] The computer-readable medium of claim 33 wherein said validation program is further structured to control said computer to perform the following additional function:

doing a partial validation each time an element is added to said formal language specification to check for completeness and correctness and mark the portion of said formal language specification just added as invalid if an error is found so that a request to correct said error can be presented later when a full validation of said formal language specification is requested.

36. [new] The computer-readable medium of claim 33 wherein said validation program is further structured to control said computer to perform the following additional functions:

checking to ensure that all elements in said Conceptual Model have a set of properties that exist and have a valid value;

performing step B by using a predetermined process and grammar for every type of formula in said Conceptual Model to ensure each is syntactically and semantically correct.

37. [new] The computer-readable medium of claim 29 wherein said validation program is structured to control said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas, and wherein a predetermined process for validation and a set of rules of grammar exist for each type of formula and a validation process and a set of rules appropriate for the type of formula being validated is used for validation of each formula.

38. The computer-readable medium of claim 33 wherein said validation program is structured to control said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	
static and dynamic integrity constraints	

derivations and valuations	
	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

and wherein said validation program is further structured to control said computer to perform the steps of:

presenting dialog boxes via a user interface by which a user may enter said formulas during a process of defining said Conceptual Model;

and wherein step B is performed by preventing a user from leaving a dialog box being used to define a formula until said formula being defined is syntactically and semantically correct.

39. The computer-readable medium of claim 33 wherein said validation program is structured to control said computer to perform the following steps:

presenting user interface tools by which a user may define said Conceptual Model and make changes thereto;

checking all affected formulas each time a change is made to said

Conceptual Model;

if the change affects a strictly validated property, then the change is rejected if the property is not given a valid value, otherwise the change is accepted;

if the change affects a property which is not strictly validated, then the user is informed should any error arise, but allowed to do the modification if he or she wishes;

if there are no affected formulas, modifying the Conceptual Model as specified by the user.